DII COE Graceful Shutdown

A Proposal

Jay Scarano
Air Force DII COE Chief Engineer
ESC/DIE (MITRE)

13 April 2001

Motivation

- Applications that get "killed" while in an inconsistent state may have trouble re-starting, or worse, cause safety problems

- No uniform mechanism exists for segment developers, system integrators or system administrators to turn off processes gracefully
  - Managing groups of segments is difficult

- Need common approach for segment-specific startup and shutdown procedures in both non-RT and RT environments
  - Codify 'common practice' in Unix programming
  - Select 1 from several commonly-used Unix/POSIX signals (SIGQUIT, SIGTERM, SIGHUP, etc)

- Opportunity to drive industry standards to meet DoD

Definition

The mechanism by which a DII COE compliant process or set of processes are notified to conclude work and clean up, prior to being terminated by the operating system

- Prior to shutting down the entire system; or
- For reconfiguration actions without forcing a reboot (e.g. "remove this segment" or "restart this application")

Applies ONLY to DII COE processes, NOT operating system processes

Proposed Requirements (Summary)

- Processes (known to DII COE) register in a list as they start up
  - Segment name, list of process IDs, time limit, priority (User ID implicit parameter, obtained from UserID of registering process)
- Segment can be shutdown by name, by process ID or by "owner" (User ID)
- Groups of segments can be shutdown by 'priority'
- All registered segments can be shutdown
- Before system shutdown
  - Each process in the list is sent a "standard" signal to terminate (with a time limit)
  - If the process hasn't completed by the end of the time limit it will issued a "hard" kill signal

Impacts to Systems

- **For legacy/heritage -- <u>none</u>**
  - **That cannot afford even minimal cost to change/adopt**
    - **The default behavior is to kill the process--precisely the <u>status quo</u>**
  - **That do not implement any particular or defined shutdown behavior**
    - **The default behavior is to kill the process--precisely the <u>status quo</u>**
  - **That implement segment-specific shutdown behavior using this approach**
    - **<u>Low cost</u> method of "wrapping" that will make segment-specific behavior consistent with COE-defined behavior**
- **For future development -- <u>minimal</u>**
  - **This service establishes good practice, facilitates integration, and enhances potential for software re-**

Current Status
Basis for Implementation

What we have:

- Documented requirements
- A POSIX-conforming reference implementation in Ada
- A method for wrapping legacy applications

What we still need:

- Select the "signal"
- Develop/vet criteria for I&RTS
- Implementation for Win2K/NT
- UNIX "C" implementation
- Reference implementation segmented
- Usage document (beyond primitives)
- er

The hard work has been done!
Let's move towards completion.

Recommendations

- Continue on Army implementation track (with RTAG support)
  - Revisit detailed requirements to ensure all Use-Cases defined and met; propose changes as appropriate
  - Consider security aspects of current and proposed mechanisms
  - Segment for RT kernel (near term)
- Adopt Graceful Shutdown for 5.x DII COE Kernel
  - Turn Army reference implementation over to DISA
  - Add to I&RTS as optional feature with phase-in schedule to make mandatory

Backups

Detailed DII COE Graceful Shutdown Requirements (from RT Extensions Kernel SRS) (1/3)

- Graceful shutdown applies only to DII COE processes
- Graceful shutdown is a 3-step process:
  1. Send 'request to shut down' signal to process
  2. Wait user/developer defined grace period
  3. Send OS immediate termination signal to process
- This ensures that processes have some time to clean up, but after step 3 they are guaranteed (by the OS) to be gone

Detailed DII COE Graceful Shutdown Requirements (from RT Extensions Kernel SRS) (2/3)

- Segments register a shutdown handler
  - Segment Name, Grace Period, Priority, List of processes to receive shutdown signal
  - Caller's UserID implicit parameter
- Shutdown can be invoked several ways:
  - By (registered) segment name
  - By specific Process ID
  - By UserName (shuts down all segments registered by that user)
  - By Priority (shuts down all segments at given priority value)
  - Shut down all registered segments

Detailed DII COE Graceful Shutdown Requirements (from RT Extensions Kernel SRS) (3/3)

- **Requires both API and CLI**
  - **CLI must conform to DII COE conventions**
  - **API in appropriate DII COE languages**
- **Caller must not block when invoking shutdown**
  - **Call to API returns immediately, shutdown occurs in background process**
- **Log appropriate events, particularly when a process is not 'dead' after grace period expires**

Wrapping Legacy Segments

```
#! /bin/sh
# Sample wrapper shellscript for Graceful Shutdown.

# register this shellscript as the handler for the
   segment
shutdown_cli register segmentname 6.0 1 $$

# segment startup commands go here
segment-specific-startup-actions

# assumes selected Shutdown Signal is SIGQUIT (3)
trap "segment-specific-shutdown-actions" 3

while (:)
do
  # sleep value should be less than 'grace period' value
  sleep 5
done
```